Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2009

Prof. Dr. Rainer Stiefelhagen
Dr. Edgar Seemann

Institut für Anthropomatik
Universität Karlsruhe (TH)

http://cvhci.ira.uka.de
rainer.stiefelhagen@kit.edu
seemann@pedestrian-detection.com
Programming

Assignments

WS 2009/10

Dr. Edgar Seemann

seemann@pedestrian-detection.com
## Termine (1)

<table>
<thead>
<tr>
<th>Termine</th>
<th>Thema</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.10.2009</td>
<td>Introduction, Applications</td>
</tr>
<tr>
<td>23.10.2009</td>
<td>Basics: Cameras, Transformations, Color</td>
</tr>
<tr>
<td>26.10.2009</td>
<td>Basics: Image Processing</td>
</tr>
<tr>
<td>30.10.2009</td>
<td>Basics: Pattern recognition</td>
</tr>
<tr>
<td>02.11.2009</td>
<td>Computer Vision: Tasks, Challenges, Learning, Performance measures</td>
</tr>
<tr>
<td>06.11.2009</td>
<td>Face Detection I: Color, Edges (Birchfield)</td>
</tr>
<tr>
<td>09.11.2009</td>
<td><strong>Project 1: Intro + Programming tips</strong></td>
</tr>
<tr>
<td>13.11.2009</td>
<td>Face Detection II: ANNs, SVM, Viola &amp; Jones</td>
</tr>
<tr>
<td>16.11.2009</td>
<td>Project 1: Questions</td>
</tr>
<tr>
<td>20.11.2009</td>
<td>Face Recognition I: Traditional Approaches, Eigenfaces, Fisherfaces, EBGM</td>
</tr>
<tr>
<td>23.12.2009</td>
<td>Face Recognition II</td>
</tr>
<tr>
<td>27.11.2009</td>
<td>Head Pose Estimation: Model-based, NN, Texture Mapping, Focus of Attention</td>
</tr>
<tr>
<td>30.11.2009</td>
<td>People Detection I</td>
</tr>
<tr>
<td>04.12.2009</td>
<td>People Detection II</td>
</tr>
<tr>
<td>07.12.2009</td>
<td><strong>Project 1: Student Presentations, Project 2: Intro</strong></td>
</tr>
<tr>
<td>11.12.2009</td>
<td>People Detection III (Part-Based Models)</td>
</tr>
<tr>
<td>14.12.2009</td>
<td>Scene Context and Geometry</td>
</tr>
<tr>
<td>18.12.2009</td>
<td>Stereo and Optical Flow</td>
</tr>
<tr>
<td>21.12.2009</td>
<td>TBA</td>
</tr>
</tbody>
</table>
This Lecture

- Student presentations

- Short Intro into Assignment 2
  - Data Sets
  - SVM library
  - HOG descriptor
Gruppen

- Gruppe 3
  - Tingyun, Zhang
  - Ning, Zhu
- Gruppe 4
  - Andreas, Wachter
  - Sina, Martens
  - Sijie, Shen
- Gruppe 8
  - Alexander, Herzog
  - Stefan, Bürger
  - Jonathan, Wehrle
- Gruppe 9
  - Marc, Essinger
- Gruppe ?

- Gruppe 5
  - Christian, Wittner
  - Matthias, Steiner
  - Christoph, Weber
- Gruppe 6
  - Nils, Adermann
  - Jan, Issac
- Gruppe 7
  - Alexander, Wirth
  - Dimitri, Majarle
  - Paul, Märgner
- Gruppe 1
  - Patrick, Grube
  - Benjamin, Hohl
  - Bastiaan, Hovestreydt
- Gruppe 2
  - Sebastian, Bodenstedt
  - Michael, Heck
Who wants to go first?
Assignment 2
Assignment 2

- People Classification
  - Decide for a given image, if it contains a person or not
  - It’s assumed that:
    - the person is centered
    - the person covers the complete image

- That is:
  1. We have to compute an appropriate feature description
  2. We have to train a binary classifier (e.g. SVM)
Training Data

- Training Set (PersonTrain.tar.bz2):
  - 2418 positive examples
  - 2436 negative examples
  - 96x160 pixels (64x128 + larger border)

- Idl-files:
  - Pos.idl
  - Neg.idl
Test Set

- Test set (PersonTestClassification.tar.bz2):
  - 1132 positive images
  - 4530 negative images
  - 70x134 pixels (64x128 + small border)
  - Ground-truth defined in testset.idl
Your Task

- Compute an .idl file, which specifies for each test image the probability of being a person
  - i.e. specify a 64x128 pixel rectangle in the image center

- Annotool helps to display results at different confidence levels
Quantitative Evaluation

- For the evaluation, we have two Python scripts
  - Directory: evaluation
    - `./fpr-rec-person.py` testset-groundtruth.idl result.idl
    - Computes true positive and false positive rate for all thresholds and writes it to plotdata.txt
  - Directory: plotting
    - `./plotSimple.py` ../evaluation/plotdata.txt
    - Plots the results from plotdata.txt
SVM libraries

- Original source and documentation:
  - SVMlight
  - libSVM
  - Okapi
  - OpenCV
SVM parameters

- Parameters
  - Kernel: e.g. linear, polynomial, rbf
  - Type: e.g. classification, regression
  - Cost ratio/weight: ratio negative/positive training examples
  - C: trade-off between training error and margin (slack variable)
Feature Description

- HOG (Histogram of Oriented Gradients) [Dalal&Triggs’05]
  - -> as described in the lecture

- Steps:
  - Compute image gradients (magnitude and orientation)
  - Build gradient histograms of local areas
    - Use interpolation to avoid quantization artifacts
OpenCV

- OpenCV is an open source computer vision library containing a large number of existing algorithms
  - Image Processing:
    - Edge Detection
    - Interest Point Detection
    - Morphological Operations
  - Machine Learning
    - SVM
    - Boosting
  - Optical Flow
  - Stereo Computation
  - Tracking
Okapi

- Please use Okapi
  - Has nice C++ API (unlike OpenCV)
  - For the API look in /usr/local/include
Images as matrices

- **Load image**
  
  ```
  Mat img = imread(image_fname);
  ```

- **Load image as grayscale image**
  
  ```
  Mat maskimg = imread(mask_fname, CV_LOAD_IMAGE_GRAYSCALE);
  ```

- **Access image elements**
  
  ```
  int pixelValue = img.at<uchar>(y,x);
  Vec3b& pixelValue = img.at<Vec3b>(y,x);
  int red = pixelValue[2];
  int green = pixelValue[1];
  int blue = pixelValue[0];
  ```
STL

- The C++ Standard Template Library provides many useful functions/classes/containers etc.
- Documentation can be found at http://www.sgi.com/tech/stl
- Examples:
  - std::vector
  - std::sort
  - std::search
- Tip: “using namespace std;” avoids the additional typing of std:: (only do this in .cpp files)
Containers

- Vectors
  - Provide dynamic arrays
  - Example:
    - `std::vector<int> numbers; //create vector of integers`
    - `numbers.push_back(5); //add 5 to vector`
    - `int val = numbers.back();`
    - `int val = numbers[0]; // array style access`

- Iterators are a generalization of pointers
  - Example:
    - `std::vector<int>::iterator it;`
    - `for (it=numbers.begin(); it!=numbers.end; ++it)`
      `std::cout << *it;`
Sorting

- **Sorting**
  - `std::vector<double> numbers;
    ...
    std::sort(numbers.begin(), numbers.end());`

- **Comparators/Functors**
  - `class compMag : public binary_function<double, double, bool>
    {
      bool operator()(double x, double y)
      {
        return fabs(x) < fabs(y);
      }
    };
    std::sort(numbers.begin(), numbers.end(),
               compMag());`
Visualization

- Visualization often helps to understand what your code is doing (and what it is doing incorrectly)
- Possibilities:
  - Write a GUI
  - Render an image and store it to disk
  - Write data to a file and use some other tool to visualize them
Things to hand in

  - Presentation (ppt<=2003, pdf, odp)
  - Source code
  - Time spent for each person in the group
End of Lecture